



<p>HORIZON 2020</p> 		<p>Deliverable ID:</p> <p>D5.6</p>	<p>Preparation date:</p> <p>17 June 2020</p>
 <p>Terahertz based Ultra High Bandwidth Wireless Access Networks</p>		<p>Milestone: Final Proposed</p>	
		<p>Title:</p> <p>Final NETWORK Layer and Management Model</p>	
		<p>Editor/Lead beneficiary (name/partner):</p> <p>Sean Ahearne / Dell EMC</p>	
		<p>Internally reviewed by (name/partner):</p> <p>Luis Gonzalez Guerrero/UCL Marco Garcia Porcel/VLC</p>	
		<p>Approved by:</p> <p>Alan Davy/WIT</p>	
<p>Dissemination level</p>			
PU	Public	X	
CO	Confidential, only for members of the consortium (including Commission Services)		

Revisions				
Version	Date	Author	Organisation	Details
0.0	20.01.20	Sean Ahearne	Dell EMC	Structure Created
0.1	24.02.20	Sean Ahearne	Dell EMC	First Draft (Intro, Overview)
0.2	16.03.20	Sean Ahearne	Dell EMC	Second Draft (Background)
0.3	13.04.20	Sean Ahearne	Dell EMC	Third Draft (Sections 3 and 4)
0.4	18.05.20	Sean Ahearne	Dell EMC	Fourth Draft (Sections 3 and 4 cont.)
0.5	02.06.20	Sean Ahearne	Dell EMC	Final Draft for Review

Table of contents

Table of contents.....	1
List of figures	2
Executive summary.....	3
1 Introduction	1
1.1 Summary	1
1.2 Structure of this document	1
1.3 Relationships with other deliverables.....	1
1.4 Acronyms and abbreviations	2
2 Overview	3
2.1 Objectives	3
2.2 Background	4
2.3 Testing Tools	5
2.4 Assumptions.....	5
3 Integration Frameworks for THz Wireless Links in a Software Defined Network	6
3.1 Hardware Control System for THz Wireless Links	6
3.2 OpenFlow Implementation	11
3.3 P4 Implementation	13
3.4 NETCONF	15
3.5 Summary	18
4 THz Network Function Virtualization	19
4.1 Topology Discovery	19
4.2 THz Routing, Load Balancing, & Fail-over.....	20
5 Conclusions	23
References	24



List of figures

Figure 2.1: SDN Network architecture	5
Figure 3.1: Internal configuration of an optical SFP module	6
Figure 3.2: Overview on an I ² C controller, typically found on a network switch	7
Figure 3.3: Contents of an SFP modules EEPROM memory blocks.	8
Figure 3.4: Excerpt from SFF-8690 showing the definition of registers for controlling wavelength.....	9
Figure 3.5: Overview of OpenFlow Architecture	11
Figure 3.6: The addition of a struct to control optical wavelength in OpenFlow 1.4	12
Figure 3.7: Example possible OpenFlow struct for THz parameters	13
Figure 3.8: P4 Architecture Model	14
Figure 3.9: Example of how to access registers as extern values in a P4 program.....	14
Figure 3.10: Example definition of P4 THz extern value.....	15
Figure 3.11: NETCONF system architecture	16
Figure 3. 12: NETCONF Architecture to Implement THz links	16
Figure 3.13: Example YANG model containing THz link parameters	17
Figure 4.1: Ryu SDN Controller Architecture	19
Figure 4.2: Example LLDP sequence diagram for THz links	20
Figure 4.3: Network topology for THz NFV testing.....	21
Figure 4.4: Excerpt of Ryu Thz Routing algorithm	21
Figure 4.5: TCP test showing re-routing of THz traffic through loss of throughput	22
Figure 4.6: Latency test result for THz fail-over function.....	22

List of tables

Table 1: Example of some possible register definitions for THz wireless links	10
Table 2: Comparison of KPI's for implementing SDN control of THz links	18



Executive summary

The deliverable D5.6 is split into two major chapters both relating to the implementation of Terahertz (THz) wireless links within a datacentre environment at the logical Network layer. Section 3 details the possible integration frameworks for the parameters required to effectively operate THz links in a datacentre environment. This involves definition of a low-level THz parameter storage system, based on an evolution of an existing manufacturer standard for the operation of optical links. This is followed by the definition of network architecture capable of utilizing this system, based on existing network protocols used in Software Defined Networking (SDN). SDN control of THz links is outlined with three of the major SDN protocols in use today: OpenFlow, P4, and NETCONF. How each protocol communicates with and controls THz links is outlined, with each protocol compared and contrasted to highlight their performance and features. Section 4 describes the creation of Virtual Network Functions (VNF's) designed to effectively utilize a network of SDN-enabled THz links in conjunction with a traditional copper and optical datacentre network. VNF's such as wireless fail-over and latency-optimized routing are described at an architectural level, followed by integration of the VNF's with an SDN controller to demonstrate functionality using network emulation. Successful demonstration of this functionality proves the ability of THz wireless links to be capable of full integration at all layers (physical, data link network) with SDN networks and automated controllers. This presents the potential for THz links to create a paradigm shift in network topology and architecture, as SDN controllers now have the ability to re-configure datacentre networks with THz links dynamically based on performance requirements. A feat previously not possible due to the fixed nature of wired copper and optical links.



1 Introduction

1.1 Summary

This deliverable aims to define and demonstrate the logical integration of Terahertz (THz) wireless links as part of a Software Defined Network (SDN). To achieve this, the creation of a network architecture capable of controlling THz links using an SDN network protocol needs to be created. This entails the definition of a hardware system capable of processing and controlling the various Physical Layer (PHY) parameters necessary for THz links to perform effectively. This system needs to be accessible using an SDN networking protocol through an Application Programming Interface (API). Once the SDN protocol has access to these parameters, an autonomous SDN controller can use these parameters to perform various Virtual Network Functions (VNF's), provided the VNF has been programmed to process the THz wireless parameters during operation. Thus, the process of this deliverable is structured as follows:

1. The definition of a THz hardware system to process and modify THz physical parameters
2. Integration of this hardware system with SDN protocols
3. Creation and implementation of SDN VNF's based on these protocols.

By doing this, we can verify full compatibility of THz wireless devices with datacentre networks.

1.2 Structure of this document

This document is structured as follows:

- Chapter 2: Overview
 - This chapter will give a summarized explanation of the objectives of this deliverable. It will also provide a background on the motivation for this deliverable, and any assumptions made about THz link performance and operation.
- Chapter 3 Integration Frameworks for THz Wireless Links in a Software Defined Network
 - This chapter covers the creation and operation of a THz hardware system capable of operating with datacentre network devices, followed by the architectural implementation of SDN protocols which can utilize it.
- Chapter 4: THz Network Function Virtualization
 - This chapter goes into detail about the implantation of automated VNF's designed around the use of the newly available THz wireless links as part of a datacentre network.
- Chapter 5: Conclusions
 - This chapter presents a conclusion based on the information gathered for this deliverable. It also discusses future possibilities that can be made to THz links for future network development.

1.3 Relationships with other deliverables

This deliverable is the third deliverable of work package 6. The work presented in this document relates to the following deliverables:

- D2.3 – Final Requirements and Scenario Specifications
 - The output of this deliverable serves to demonstrate and achieve requirements necessary for effective use of THz wireless links in a datacentre environment. These necessary requirements are outlined in D2.3.
- D5.2 & D5.4 - (Final PHY/DLL Layer Models & Simulations)
 - A global THz physical parameter list was created outlining the possible physical capabilities of TERAPOD THz hardware devices. This list was



unified between all WP5 deliverables to improve consistency between layers and deliverables.

- D6.6 – Final Datacenter Demonstrator
 - Where possible, architectural and functional outputs from this deliverable will be considered as an input to D6.6 for real-world demonstration.

The following partners have contributed to this deliverable:

- Sean Ahearne (Dell EMC)
- Niamh O'Mahony (Dell EMC)
- Luis Gonzalez Guerrero (UCL)
- Nouredine Bounjah (TSSG)
- Saim Ghafoor (TSSG)
- Johannes Eckhardt (TUBS)

1.4 Acronyms and abbreviations

THz – Terahertz

OSI – Open Systems Interconnection

PHY – Physical Layer

DLL – Data-Link Layer

NET – Network Layer

SDN – Software Defined Networking

VNF – Virtual Network Function

OS – Operating System

KPI – Key Performance Indicator

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

LLDP – Link Layer Discovery Protocol

M/Gbps – Mega/Gigabits per Second

SFP – Small Form-factor Pluggable

m/μ/ns – Milli/Micro/Nano-seconds

BER – Bit Error Rate

UTC-PD – Uni-Travelling Carrier Photo Diode

SFP – Small Form-factor Pluggable

ASIC – Application Specific Integrated Circuit

EEPROM - Electrically Erasable Programmable Read-Only Memory

I²C – Inter-Integrated Circuit

API – Application Programming Interface

NETCONF – NETwork CONFiguration

YANG – Yet Another Next Generation

MIMO – Multiple Input Multiple Output

TDMA – Time Division Multiple Access



2 Overview

2.1 Objectives

The objective of this deliverable is determining the possibility of integration of THz wireless links as part of a traditional datacentre network. In order to achieve this integration, a number of questions need to be answered on how THz links can interact with all other devices typically found in a datacentre network. Software Defined Networking (SDN) plays a prominent role in modern datacentres, meaning integration of THz links using SDN principles is required¹. Achieving this integration enables a new form of communication technology to be introduced to SDN datacentres, with the vast majority of previous networks only utilizing fixed point to point wired links.

In order to achieve this objective, a method to enable datacentre network devices (such as a network switch) to monitor and control THz wireless devices needs to be created. An interface between the THz and a network switch needs to be defined. This interface needs to be compatible with the relatively simple electronics controlling the THz device, and the Operating System (OS) running on the network switch. Once this interface is defined, a method to abstract this interface such that it can be controlled using SDN principles needs to be investigated. A number of SDN protocols exist that are capable of this abstraction, provided extensions are made to the protocol to support THz links. This deliverable investigates the implementation of these extensions to three SDN protocols: OpenFlow², P4³, and NETCONF⁴.

Once the architecture of abstraction of THz link parameters is achieved, the next objective is to demonstrate the implementation of SDN Virtual Network Functions (VNF's) designed to utilize THz links. VNF's are a virtual and software-defined implementation of network features frequently used and utilized in hardware in traditional enterprise networks, such routing, load-balancing, and firewalling⁵. In a similar fashion to the previous objective, these VNF's require an extension in order to be able to control and manage THz wireless links. As these functions are software-defined, this enables the extensions to be implemented relatively quickly and without any possible hardware incompatibility issues. If the required THz parameters necessary to perform the required function have been abstracted using an SDN protocol, the VNF can quickly access and control the THz link through the protocols Application Programming Interface (API).

The demonstration of this implementation will be performed using network emulation. A number of network topologies and scenarios will be created in order to test the modified VNF's with an emulated network of THz wireless links operating in a network in conjunction with traditional wired links. For each modified VNF tested, a network scenario will occur that forces the networks SDN controller to execute that function. The emulated network will be monitored to investigate and determine if the extended VNF performed correctly with the newly introduced THz link(s). If the VNF does perform correctly, this completes this objective and provides proof that THz links can be successfully and fully integrated into a datacentre network. It also proves that SDN control of THz links is possible, thus enabling autonomous management of this links with an SDN controller. This has the potential to greatly increase the scalability and flexibility of any datacentre SDN utilizing THz links, as the network topology and functions performed can be dynamically and autonomously changed by the SDN controller based on the performance and service requirements requested by users and application on the network.



2.2 Background

The motivation for this research stems from the investigation and creation of wireless devices capable of operating in the THz frequency spectrum. These devices are capable of data transmission, thus could potentially be feasible for use in network environments with high data rate requirements⁶. One of these environments is a datacentre, where data rate and reliability requirements exceed what is currently possible with traditional wireless technologies such as WiFi. Modern datacentres typically operate at data rates at a minimum of 10 Gigabits per second (Gbps) per link, scaling up to data rates up to and beyond 400Gbps⁷. In contrast, state of the art WiFi (802.11ax) cannot reach half of the minimum bitrate required, with realistic throughput often lower than specified⁸.

This changes with the introduction of THz wireless links based on Uni-Travelling Carrier Photodiode (UTC-PD) technology however⁹. These devices are capable of achieving the data rates required in datacentre communication, with laboratory tests of these devices achieving data rates of 100Gbps¹⁰. This presents the potential for adoption of these new types of wireless links into a datacentre environment. In order for this to occur, a number of challenges must be overcome to integrate these devices with datacentre network devices.

SDN is a key technology implemented in modern datacentre networks, enabling separation of a modern networks control plane from the packet processing data plane. This separation allows network function to be virtualized into software, and data plane devices created by different manufacturers to be controlled and managed inter-operably using a common API. The general architecture of SDN can be seen in Figure 2.1. Data plane devices such as network switches enable connectivity between all devices on the network using a variety of physical connections. In this example, the addition of THz links to the network is shown as a dotted red line alongside traditional copper connections in black, and optical connections in blue.

To achieve SDN control of THz links as well as advanced THz VNF's, the data plane devices must be capable of passing information about the physical parameters of the THz links to an SDN controller via an API. Once the controller has access to the THz parameter information, VNF's can be created for using that information to enable network-level support for advanced THz features such as beam steering. Once these THz VNF's have been standardized they can be integrated with a network management plane software, maintaining a heterogeneous network of copper, optical and THz links using its available VNF's. The operation of this VNF's and SDN controller will be defined by service-level agreements, which define what Key Performance Indications (KPI's) are desired by the users of the network. This in turn will impact how the SDN controller autonomously manages its network of links in order to achieve the desired KPI's.

The integration of THz links with software defined networks will pave the way for adoption of the technology in datacentre environments. The ability for THz links to be autonomously configured in real-time with future advanced features such as Multiple Input Multiple Output (MIMO), beam steering/switching, and Time Division Multiple Access (TDMA) could present a drastic change in the network architecture of future datacentre deployments. The increased flexibility offered by THz links over their copper or optical counterparts could enable future THz datacentre designs to utilize all possible available network bandwidth provided by the underlying infrastructure, something previously impossible with wired links. This combined with the ability to modify available bandwidth between devices within the infrastructure in real-time based on user requirements presents a strong argument for continued research and development for wireless datacentre links.



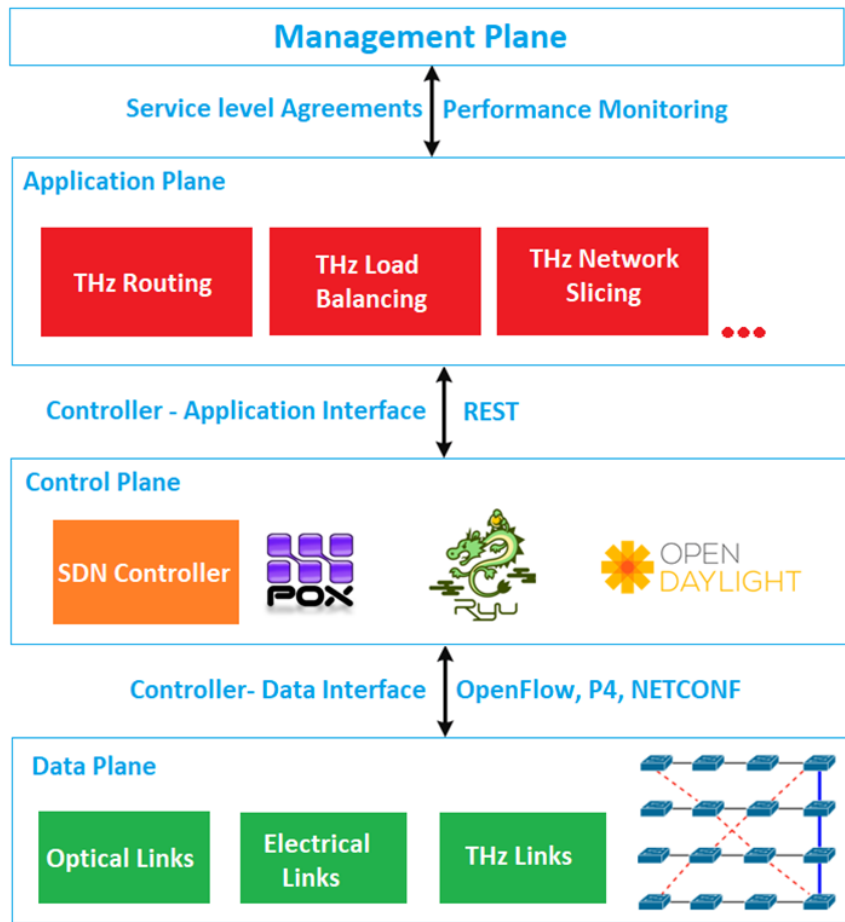


Figure 2.1: SDN Network architecture

2.3 Testing Tools

The following lists the tools that have been used for the testing and results gathering described in this deliverable.

1. iPerf3
 - a. Simple traffic generator¹¹
2. Wireshark
 - a. Packet capture & analytics tool¹²
3. Mininet
 - a. Network Layer Emulator¹³

2.4 Assumptions

In order for development of network layer protocols to continue for THz links a number of assumptions must be made. In section 3 the development of a microcontroller used to control THz devices is described, followed by the definition of SDN protocols based on this architecture. For section 4, it is assumed that this architecture has been already created for the purposes of implementing THz network functions. As microcontroller development for THz devices will not occur until much later in the commercialization process, the THz network functions have been developed using simulated values the SDN controller would receive from the microcontroller. These parameters are simulated within the range described by the global parameter list which specifies the physical device parameters and performance achieved by THz devices in the TERAPOD project.



3 Integration Frameworks for THz Wireless Links in a Software Defined Network

3.1 Hardware Control System for THz Wireless Links

The first step towards full-integration of THz wireless links as part of a Software-Defined network is development of an electronic system which can monitor and control the various physical-layer operations performed in THz wireless communication. For a THz link to be useful in data communication a number of physical parameters such as wireless frequency, power, noise, and more need to be known. These types of parameters must be monitored by various sensors placed throughout the THz system in order to ensure the hardware is operating within specifications. As the development and commercialization of THz devices continues, the creation of an Application Specific Integrated Circuit (ASIC) solely designed for management of a THz wireless system will be developed. This type of ASIC is often developed using a simple micro-controller, with basic functions and firmware developed to prevent damage to underlying THz hardware and maintain stable operation¹⁴. In datacentre networks, this type of ASIC design already exists to assist in the operation of laser-based optical links. The Small Form-factor Pluggable (SFP) standard is a widely used method in datacentres to insert various types of communication equipment into network devices¹⁵. These SFP modules primarily operate using copper or optical cables, with various levels of bitrate and range available depending on user requirements. In a laser-based optical SFP module, a micro-controller ASIC exists within the packaged module that automatically monitors and maintains operation of the laser according to specifications set by its manufacturer. Accompanying this ASIC is a block of Electrically Erasable Programmable Read-only Memory (EEPROM). This memory is used by the ASIC to store values recorded by connected hardware sensors such as the lasers temperature, power consumption, wavelength and more. An example of the configuration of a laser SFP module can be seen in Figure 3.1, with the microcontroller ASIC and EEPROM memory located in the “Diagnosis” block.

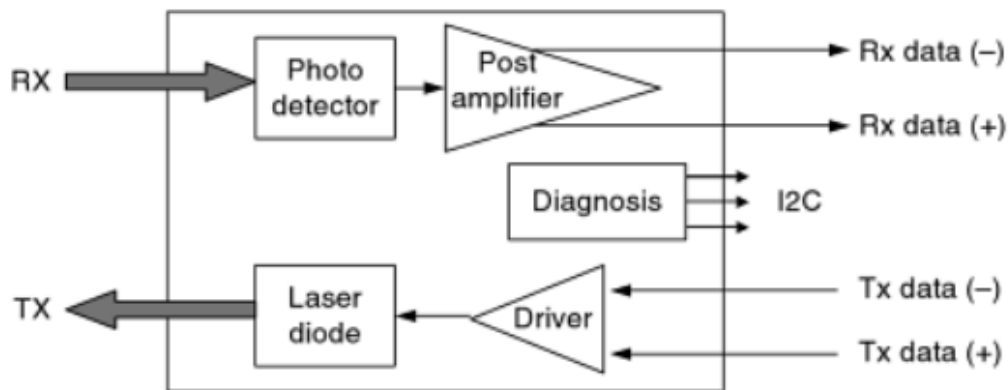


Figure 3.1: Internal configuration of an optical SFP module¹⁶

While this ASIC can provide robust and stable operation of the hardware components inside its SFP module, it would be beneficial for the sensor data stored in EEPROM to be accessible to the host device the module is plugged in to. This would allow the host device to also monitor hardware performance and notify a network administrator with a warning if an SFP module is overheating or has encountered some other type of error. To facilitate communication between the host device and SFP module, an electrical communication bus is required. The Inter-Integrated Circuit (I²C) bus is the accepted standard to enable this communication¹⁷. This bus provides a two-wire (serial clock and serial data) limited electrical interface between the host device and the modules EEPROM.



As datacentre network switches typically contain several SFP ports, an I²C controller is often used to simplify access to each SFP modules EEPROM from the host devices perspective. An example of the operation of this controller can be seen in Figure 3.2, where the I²C bus master is the Central Processing Unit (CPU) of a network switch in this instance.

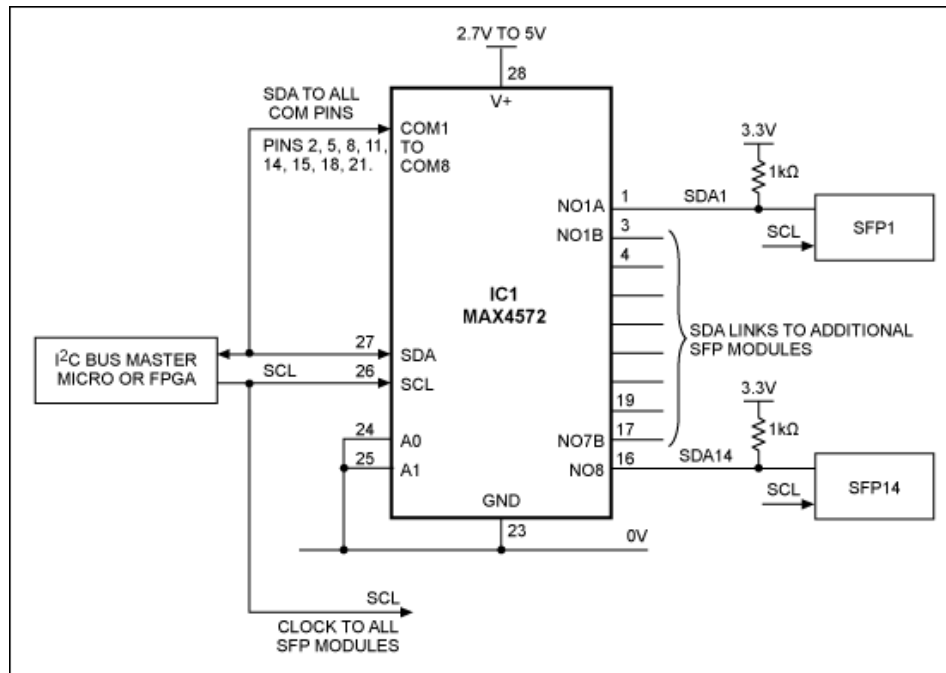


Figure 3.2: Overview on an I²C controller, typically found on a network switch¹⁸

In order for a network switch to correctly read and interpret the sensor data contained in an SFP's EEPROM, the switches Operating System (OS) must contain the appropriate software to support interaction with an I²C bus. The majority of modern datacentre switches run an OS based on a Linux Kernel, with driver support for I²C communication immediately available¹⁹. Software can be written utilizing this driver to access and read the EEPROM memory of an SFP module. Values can also be written in certain circumstances, with the microcontroller executing a function to perform on a memory write event. The parameters and values stored in an SFP EEPROM block are standardized. A Multi-Source Agreement (MSA) exists between all major manufacturers of SFP modules to store certain information and sensor values in specific memory locations. The SFF-8472 specification details the use of two addressable EEPROM memory blocks accessible via the I²C data bus²⁰. One of these blocks is known as the information block, containing details about the SFP modules manufacturer, serial number, and other product information. The second diagnostic memory block contains sensor information about the optical hardware within the module. These memory blocks are arranged in a series of registers. Each register contains two hexadecimal digits, the product of which represents a decimal value range from 0 to 255 (1 byte). Both memory blocks contain 256 registers, split into 128 block "pages". The SFF specification details what information is stored in each register location, with some values represented as a product or function of two or more register values. To correctly interpret the information stored in these registers, a piece of software running on the switch's OS must read the memory blocks using the I²C bus and decode the values it receives based on the SFF specification. When writing a value to EEPROM, if input is given in decimal form or another format it must be re-calculated and re-encoded to the correct hexadecimal format required. A "raw" memory dump of both memory blocks can be seen in Figure 3.3, this particular memory dump being from a 1550nm SFP optical transceiver module. Note the large number of un-used "00" registers, suggesting available space for future specification expansion.



```

[+] INFO
03 04 07 00 00 00 00 00 00 00 00 06 67 00 50 FF
00 00 00 00 4A 44 53 55 20 20 20 20 20 20 20 20
20 20 20 20 00 00 01 9C 57 52 54 2D 53 46 50 50
54 30 31 35 53 43 20 20 30 30 30 30 06 15 00 1B
06 5A 0A 04 46 45 34 38 35 35 37 38 30 31 30 34
20 20 20 20 31 34 31 31 32 36 20 20 68 F0 05 4F
31 A9 4A 44 53 55 4E 41 20 20 20 20 20 20 20 20
53 46 50 50 54 30 31 35 53 43 FF FF FF FF FF FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

[+] DIAG
49 00 F8 00 46 00 FB 00 8D CC 74 04 87 5A 7A 75
06 D8 1D 4C B9 8C 30 D4 4D F0 13 93 3D E8 18 A5
0F 8D 00 0C 09 CF 00 13 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00
01 00 00 00 01 00 00 00 01 00 00 00 00 00 00 DF
24 3A 7C 5D 4A 43 27 10 32 7D 00 00 00 00 00 00
00 80 00 00 00 80 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

Figure 3.3: Contents of an SFP modules EEPROM memory blocks.

The SFF-8472 specification has been updated numerous times since its inception, with the current revision version number being 12.3. Separate to this, a number of other SFF specifications have been created to expand the functionality of optical SFP modules in significant enough ways to warrant a new specification. One of these specifications is SFF-8690, which standardizes the method to interact with and control optical SFP modules with tuneable laser wavelengths²¹. This specification achieves the implementation of this feature through the use of memory paging. On a tuneable SFP module, the diagnostic EEPROM block contains a third block of 128 memory registers. This third memory block is accessed by writing “02” to the 127th register of the first block, which triggers the microcontroller to change the information contained in the second-half of the EEPROM block to page 2, referring to the new register definitions created in SFF-8690 for tuneable SFPs. Having the new memory page accessible using this method allows these modules to be backwards compatible with existing SFP network devices, though without tuneability functionality. It may be possible for this functionality to be added to existing network devices with a software update, though some devices may not feature the EEPROM write functionality required. An excerpt from the SFF-8690 specification can be seen in Figure 3.4, showing some of the newly defined registers and what their values represent. Note that the manufacturer is given multiple options for which type of measurement unit they wish to use and store. Different unit types can be stored at different register locations, with flag registers defined to identify which particular units and mode the manufacturers microcontroller operates with. As there are a number of different possible unit systems and modes of operation of tuneable optical modules, all these different possibilities must be accounted for in the software of any network device and OS that wishes to support them.



A2h Address	Size	Name	Description
Bytes 132 (MSB) & 133 (LSB)	2 bytes	LFL1	Lasers First Frequency (THz)
Bytes 134 (MSB) & 135 (LSB)	2 bytes	LFL2	Lasers First Frequency (GHz*10), in units of 0.1 GHz
Bytes 136 (MSB) & 137 (LSB)	2 bytes	LFH1	Lasers Last Frequency (THz)
Bytes 138 (MSB) & 139 (LSB)	2 bytes	LFH2	Lasers Last Frequency (GHz*10), in units of 0.1 GHz
Bytes 140 (MSB) & 141 (LSB)	2 bytes	LGrid	Laser's minimum supported grid spacing (GHz*10), i.e., in units of 0.1 GHz NOTE: LGrid can be a positive or negative number.

Figure 3.4: Excerpt from SFF-8690 showing the definition of registers for controlling wavelength

With the hardware-level operation of optical SFP modules now known, the applicability of this system to THz wireless links is evident. Development of THz wireless links based on the principles of the SFP specifications is a logical way of implementing THz device compatibility with standard datacentre network equipment, where the SFP standard is already ubiquitous. Like SFF-8690, we propose a similar extension which enables a new memory page containing all physical parameters related to the monitoring and operation of THz wireless links. Once THz devices reach maturity, a microcontroller can be developed to maintain operating stability of the THz device. This microcontroller will also perform all the new functions available, with it being capable of executing these functions when a user writes a new value to one of its new EEPROM registers. An example of some of the possible register definitions can be seen in Table 1. Note that these values are for reference only. The decision on what information to store and in what registers and format can be standardized when THz devices reach a higher Technology Readiness Level (TRL). Even after an official specification is defined, these specifications can still be revised to add new THz functions in future. Expanding on some of the example definitions provided in the table, the WF1 variable refers to the centre frequency of the THz link. This variable is two bytes in size, represent a maximum possible decimal value range from 0 to 65535. With units of 0.1 GHz, this gives a maximum possible range of 0.1 to 6553.5 GHz. While this is excessive given the current state of the art of THz devices, it leaves a very wide frequency range available for future THz devices without requiring a new revision. It must be considered that once a memory location has been defined, it cannot be changed in future revisions for backwards compatibility reasons. This means only registers which were previously un-used can be defined in future revisions. The CW1 represents channel width, with a range of 0.01 to 655.35 GHz. If reduction in memory space is required, CW1 could be reduced to a single byte value, with a range of 0.01 to 2.25 GHz, 0.1 to 22.5 GHz or 1 to 255GHz depending on requirements and device characteristics. TX power is a 4-byte value, ranging from a power output of 1nW up to 4.29W (2³²). Bitrate shares the same range as WF1, representing 100Mbps to 6.5Tbps. Bit-Error-Rate (BER) is a single byte value with a range from 1x10⁻¹ to 1x10⁻²⁵⁵. Other methods to detect and control the error rate of the active THz link can be defined and can be automatically controlled by the Physical (PHY), Data-Link, (DLL) or Network layer(NET) depending on user demands. The modulation scheme used can similarly be automatically decided by the microcontroller based on other THz device factors, or it can be “manually” selected by the DLL or NET layers by writing the desired value to the register. MCS index in this context refers to an index used in the 802.11 WiFi specifications related to modulation formats (QPSK, QAM-16 QAM-256 etc.). Alternatively, a number of the previously mentioned variables can be defined such that they are compliant with an associated IEEE specification, such as IEEE 802.15.3d-2017²². Defining variables in this fashion ensures they are compliant with the specification, while also potentially reducing the memory space required for some variables such as frequency and channel width due these variables now having a fixed range according to the specification.



A2h Address	Size	Name	Description
Bytes 132 (MSB) & 133 (LSB)	2 bytes	WF1	Wireless link frequency (GHz), in units of 0.1 GHz
Bytes 134 (MSB) & 135 (LSB)	2 bytes	CW1	Wireless channel width (GHz), units of 0.01 GHz
Bytes 136 (MSB) - 139 (LSB)	4 bytes	TX1	Wireless TX power (W), units of 10^{-9} W (nW)
Bytes 140 (MSB) - 143 (LSB)	4 bytes	RX1	Wireless RX power (W), units of 10^{-9} W (nW)
Bytes 144 (MSB) – 145 (LSB)	2 bytes	DR1	TX Bitrate (Gbps), units of 0.1Gbps
Bytes 146 (MSB) – 147 (LSB)	2 bytes	DR2	RX Bitrate (Gbps), units of 0.1Gbps
Byte 148	1 byte	BER1	RX Bit Error Rate (BER), measured as 1×10^{-X}
Byte 149	1 byte	MCS1	TX Modulation scheme, measured as MCS index

Table 1: Example of some possible register definitions for THz wireless links

In the event that more than 128 bytes of memory are required to store all the parameters necessary for THz operation, extra memory “pages” can be added as required as the majority of the 255 available pages are currently unassigned. Certain parameters can also be derived using a more complex formula instead of a simple product.

These example values represent the structure which can be defined to integrate THz wireless links with the SFP specification. A specification extension designed in this way would allow for backwards-compatibility with existing SFP network devices. Functionality would be limited for existing devices due to the lack of software functionality to inspect for and decode the new THz pages. The THz hardware can still operate and establish a THz link in this case however, as the embedded microcontroller can sweep and search for other THz links in the area and establish a point to point link if another device is found. For more advanced features such as THz beam steering or point to multi-point communication, the ability of the SFP host device to communicate with the microcontroller is required. This can be achieved on standard datacentre switch by updating its OS with new software code to read and write to the new THz EEPROM page(s). This would enable the switch and hence the network administrator to control and configure the THz link as they desire.

While user control of THz links has been established at this point, integration of THz links with SDN has not yet been achieved. At this point controlling the THz link can only be done from within the switch itself, either manually by a user or automatically via the OS. In order for the links to be considered part of a Software-Defined Network, they need to be controllable and configurable from an autonomous SDN controller. This means that the THz EEPROM values need to be abstracted such that they are accessible and able to be modified by a remote device. The use on an SDN process, protocol, and API are necessary to achieve this type of functionality. There are a number of SDN protocols available, and the implementation of new features such as THz link monitoring and control is highly varied and non-trivial for each protocol. In many cases once a specification or revision is created for an SDN protocol, it is not backward compatible with legacy devices. The reason for this is that many SDN control protocols are highly integrated into the data plane (DLL), meaning support for a specific version of that protocol is defined in the hardware of a network switches packet processing ASIC. This often means that any revision of an SDN protocol that requires an increase in memory space is unlikely to be supported by an existing packet forwarding ASIC, as the ASIC was fabricated without the required memory space. This is not necessarily true for all SDN protocols however. An investigation on how to abstract these THz parameters with three of the major SDN protocols will now be documented.



3.2 OpenFlow Implementation

The OpenFlow protocol is the original and most widely-known protocol in SDN. This pioneering protocol introduced the concept of a network with a centralized control plane. This control plane forms the basis of the “brain” of the network, controlling network functions such as routing, firewalling and load balancing. This control plane is fully software-based and can be run on any commodity server. This control plane connects to an OpenFlow supporting network switch using the OpenFlow API. This switch no longer operates using its own internal control plane, leaving only the packet-processing ASIC (known as the data plane) available to be configured. OpenFlow supporting ASIC’s contain a data path architecture that processes incoming packets through a series of flow tables containing flow rules. Each incoming packet progresses through the available flow tables until it successfully matches an existing flow rule. Each rule in a flow table has an associated action to perform when a packet is matched to it such as sending the packet out a specific port, sending the packet to an SDN controller for inspection, dropping the packet and more. A diagram of the standard OpenFlow architecture can be seen in Figure 3.5. An SDN controller connects to the OpenFlow Channel/Agent service which runs on the switch OS and installs new flow rules and actions into its packet forwarding ASIC, based on what network functions the user has chosen to implement on the network. Thus, network functions such as routing and load balancing can now be achieved on a network by installing the correct set of flow rules and actions on each device in the network. This eliminates the need for separate hardware devices to perform these actions as is required in traditional networking. It also removes the need for complex and proprietary control planes in each network device, thus enabling heterogenous network infrastructures with inter-operable OpenFlow network devices available from any supporting manufacturer. This is in contrast to traditional networking, where many control plane management systems were proprietary and specific to each manufacturer. The advantages presented by software defined network functions and commodification of network devices has led SDN and the OpenFlow protocol to become widely adopted in datacentre environments and beyond.

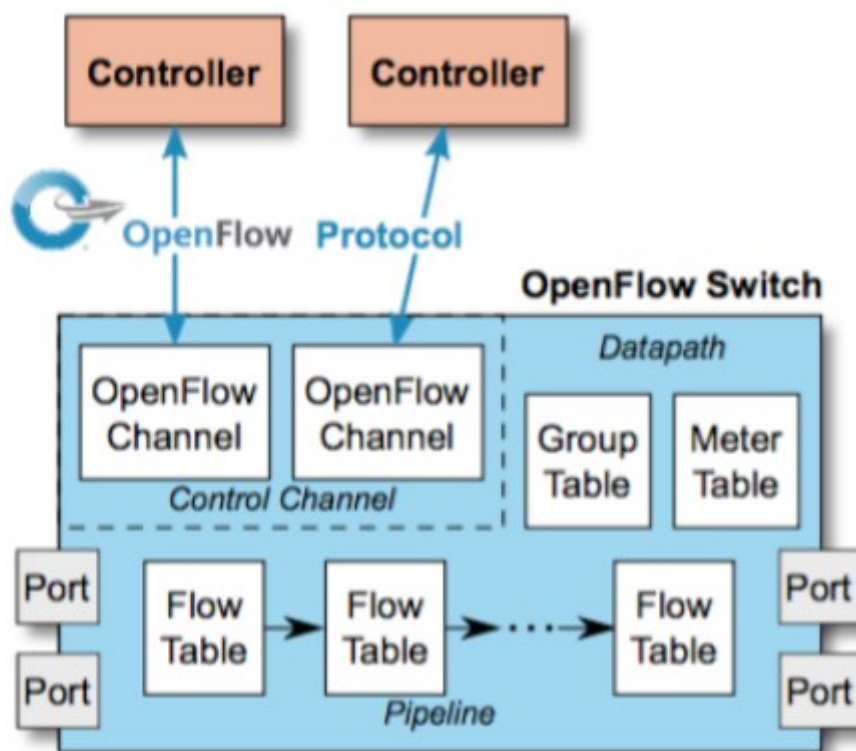


Figure 3.5: Overview of OpenFlow Architecture²³



The OpenFlow protocol has seen many major revisions since its inception, adding new and improved features for each successive revision. One major revision of the OpenFlow protocol is version 1.4, which added support for tunable optical SFP modules based on the previously mentioned SFF-8690 specification. To enable this functionality in OpenFlow, the addition of new memory blocks (also known as structs) was required. These structs play a key role in the operation of the packet forwarding ASIC, as it uses them to store and update many variables used in real-time as packets are processed through the flow table pipeline. One of these structs can be seen in Figure 3.6, specifying a number of variables which can be used to store information for the optical properties of tuneable SFP modules. It is noticeable that the size in bytes and names of these variables are not exactly the same as the SFF-8690 specification, signifying that a software translation of these variables to the specification occurs elsewhere in the process.

```
/* Optical port description property. */
struct ofp_port_desc_prop_optical {
    uint16_t      type;      /* OFPPDPT_30OPTICAL. */
    uint16_t      length;    /* Length in bytes of this property. */
    uint8_t       pad[4];    /* Align to 64 bits. */

    uint32_t supported;      /* Features supported by the port. */
    uint32_t tx_min_freq_lmda; /* Minimum TX Frequency/Wavelength */
    uint32_t tx_max_freq_lmda; /* Maximum TX Frequency/Wavelength */
    uint32_t tx_grid_freq_lmda; /* TX Grid Spacing Frequency/Wavelength */
    uint32_t rx_min_freq_lmda; /* Minimum RX Frequency/Wavelength */
    uint32_t rx_max_freq_lmda; /* Maximum RX Frequency/Wavelength */
    uint32_t rx_grid_freq_lmda; /* RX Grid Spacing Frequency/Wavelength */
    uint16_t tx_pwr_min;      /* Minimum TX power */
    uint16_t tx_pwr_max;      /* Maximum TX power */
};
OFP_ASSERT(sizeof(struct ofp_port_desc_prop_optical) == 40);
```

Figure 3.6: The addition of a struct to control optical wavelength in OpenFlow 1.4²⁴

As the addition of these new structs in version 1.4 required significant changes to both the memory and hardware functionality of OpenFlow supporting devices, it resulted in the version being incompatible with all previous versions of OpenFlow. This means that a network ASIC manufactured to the 1.3 specification is not capable of being updated to 1.4. This disadvantage has to be taken into consideration when designing and implementing OpenFlow support of THz links. The implementation of THz links in OpenFlow follows the same design principles as the implementation of support for SFF-8690. A new subsequent protocol version can be defined, and a series of memory structs can be created that mimics the design of the EEPROM parameters for THz links. An example of this THz struct can be seen in Figure 3.7. Based on the OpenFlow 1.4 specification these variables will also be independent from the associated THz SFF specification and will be transformed to the required values by a software function. As previously mentioned, this implementation will be incompatible with devices supporting previous version of OpenFlow due to memory and hardware differences. A THz SFP module could still transmit and receive data on a legacy OpenFlow switch, but visibility and control of the THz parameters and features would not be available to the SDN controller. Nevertheless, developing support of THz links with the OpenFlow protocol should be considered, as it remains the most ubiquitous SDN protocol with a large development community surrounding it. It is also the simplest method of achieving full data-plane integration of THz links using a single protocol. Its fixed hardware nature and high production volumes allow for rapid prototyping with lower development costs and likely a shorter time to a production-ready release in comparison to other SDN protocols.




```

/* THz port description property. */
struct ofp_port_desc_prop_thz {
    uint16_t      type; /* OFPPDPT_THZ */
    uint16_t      length; /* Length in bytes of this property. */
    uint8_t       pad[4]; /* Align to 64 bits. */

    uint32_t supported; /* Features supported by the port. */
    uint32_t tx_freq; /* THz TX Frequency */
    uint32_t tx_rate; /* THz TX bitrate */
    uint32_t tx_chan_freq; /* TX THz Channel Width Frequency */
    uint32_t rx_freq; /* RX Frequency */
    uint32_t rx_rate; /* RX bitrate */
    uint32_t rx_chan_freq; /* RX Channel Width Frequency */
    uint16_t tx_pwr; /* TX power */
    uint16_t rx_pwr; /* RX power */
};
OFP_ASSERT(sizeof(struct ofp_port_desc_prop_thz) == 40);

```

Figure 3.7: Example possible OpenFlow struct for THz parameters

3.3 P4 Implementation

P4 is a programming language specifically designed for use with data plane devices. P4 aims to change the way traditional packet forwarding is designed and implemented. The original specification was created in 2014, with an updated specification released in 2016²⁵²⁶. P4 follows the same principle for forwarding packets as OpenFlow as it also contains a match-action table structure. Unlike OpenFlow however, the process by which a packet is processed through the pipeline is now programmable instead of fixed in nature as it is OpenFlow. P4 achieves this by having programmable parsing logic. In a traditional network, packets sent through the network are constructed following the standard Open Systems Interconnect (OSI) model. Packets in this model contain strictly defined headers with each header identifying the protocol it is operating on, and other relevant values related to the use of that protocol. A traditional packet processing ASIC contains a fixed function pipeline where each function is designed to parse and process a specific header and protocol. OpenFlow devices also use fixed parsing logic. By using programmable parsing logic P4 network devices are protocol independent, meaning its packet processing capabilities are not limited by the OSI model or currently defined network protocols. A network built entirely of P4 devices could enable a fully software-defined data plane, where packet headers can be created, parsed and processed using a definition entirely developed by the owner of the network. This definition can also be re-configured on any device at any time after deployment, enabling new network protocols to be created and implemented on existing devices, where the creation of a new ASIC/device would be required in OpenFlow to support the new protocol. This software-defined parser also enables the P4 architecture to be compiled and implemented across multiple different device architectures while sharing the same functionality. These devices for which P4 code can be compiled for are known as targets, and can include standard CPU's, Field Programmable Grid Arrays (FPGA's), Systems on a Chip (SoC's) and ASIC's. Programmable actions are also possible with P4, which gives the user a high amount of flexibility in what to do with a packet compared to the fixed number of actions possible using OpenFlow. P4 also contains a feature known as externs, these externs are values or parameters that can be used to store information outside of the packet processing pipeline thus enabling stateful processing. The P4 architecture model can be seen in Figure 3.8.



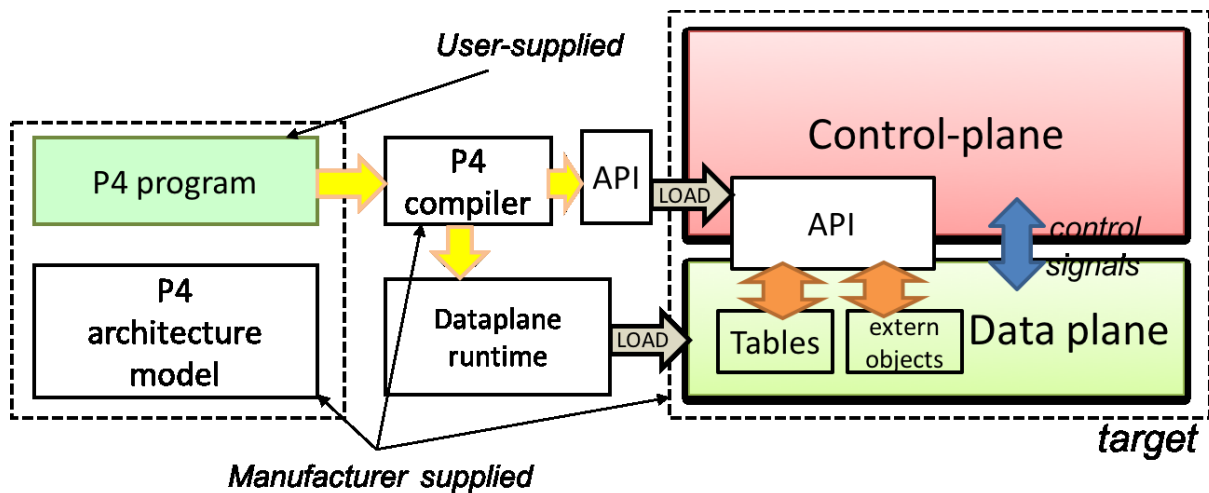


Figure 3.8: P4 Architecture Model ²⁶

A user creates their P4 program containing all their desired network functions in the form of parsing and action code. This code is created based on a manufacturer-created architecture model, which specifies the externs available on the target device and the arithmetic logic operations it is capable of performing. This program is then compiled by a P4 compiler which is also supplied by the manufacturer of the target device. The compiler loads the defined parsing and action logic into the data plane of the device and specifies the desired API to allow the control plane SDN controller to interact with the device. The SDN controller can use this API to install new rules in the match-action tables and read from and write to the available extern objects. The capabilities of P4 make it a very powerful language for datacentre networks, with the improved flexibility it brings compared to OpenFlow being considered to outweigh the disadvantage of increased complexity.

In order for THz links to be configurable by an SDN controller using P4, a number of conditions must be met. Firstly, the data plane device must contain an I²C bus in order to access the SFP microcontroller. Secondly, control logic must exist within the P4 ASIC to allow access to the THz SFP EEPROM in the form of addressable extern values. The ability for registers to be defined can be seen in Figure 3.9, showing how to initialize a bank of registers and how to read and write to a specific register. This is a standard feature defined in the P4-16 specification. The ability to define a bank of registers means that definition of a register bank that matches the proposed THz SFF specification is possible. Example P4 code to create a THz parameter register bank and associated read and write functions can be seen in Figure 3.10. There are a few things to note about this example. The first is the definition of the memory bank follows that of the EEPROM, with a bank of 128 registers (2nd page of the diagnosis block) which are 8 bits (1 byte) in size. Defining the bank in this way would allow for a direct mapping of the EEPROM registers to the P4 memory bank, thus requiring no translation function to be necessary.

```
extern Register<T> {
    Register<bit<32> size>;
    T read<bit<32> index>;
    void write<bit<32> index, T value>;
}
```

The type τ has to be specified when instantiating a set of registers, by specializing the Register type:

```
Register<bit<32>>(128) registerBank;
```

Figure 3.9: Example of how to access registers as extern values in a P4 program²⁶



It can be seen in the register definition for THz frequency however that it requires a size of 16 bits as given in the SFF specification. There are a number of solutions to this. The first is to simply modify the code to split the register definition into two separate 8-bit sections and re-combine them to get the correct value, performing the inverse for a write operation. The next solution is for the device manufacturer to define a function in their architecture model which automatically recognizes when a defined register exceeds the size of a single register in the memory bank. When the read or write function is called, it will automatically re-factor the register for the correct size of the bank based on the index value, adding 1 to the index value for every 8 bits. An indirect memory bank mapping is also possible as seen in OpenFlow, where a memory bank with 32-bit registers is defined. In this case, a function to translate the values to the SFF specification will be required in the manufacturer's architecture model. The index value of 4 in this example refers to the first register used to define THz frequency in the example SFF specification. Byte 132 is 4th byte in the new EEPROM page.

```
Register<bit<8>>>(128) THzBank;

extern Register<thz_freq> {
    Register(bit<16> size);
    thz_freq read(bit<16> index);
    void write(bit<16> index, thz_freq value);
}
//index = 4 (132 - 128)
```

Figure 3.10: Example definition of P4 THz extern value

A THz memory bank should be available for each SFP port contained on the switch. Once the extern mapping is completed, match-actions can be created that read or write to the extern registers. Programmable actions can be created which can access and modify THz link parameters in real-time on a packet by packet basis if required. This could be useful for implementing THz features such as high-speed beam steering or switching in the future. These extern values will also be accessible to the SDN controller via the chosen control plane API, where a network of P4 THz switches can be monitored by the controller and network functions can be implemented. Given the programmable nature of P4, it is also possible for new THz features added in a future SFF specification to be programmed into already existing P4 devices. Provided it has the correct architecture model to allow for changes in register definitions. The software-defined nature of controllers also allows them to be updated to support new THz features as they become available.

3.4 NETCONF

The NETwork CONFIguration (NETCONF) protocol pre-dates the era of SDN, and previously existed as a network protocol which could be used to remotely change the configuration of supporting network devices. A remote network device such as a user or SDN controller can connect to a network device such as a switch or router using the NETCONF protocol and read or update the devices current configuration. To achieve this, a NETCONF server must be ran as a service contained within the network devices OS. This NETCONF service maintains a datastore of Yet Another Next Generation (YANG) modules. These modules are used to define specific parameters that are available to be configured on the network device. When a user or controller wants to change or read the configuration of a device, it connects to the NETCONF server and sends or requests data in the same structure and format of that YANG module. An example diagram of the NETCONF architecture can be seen in Figure 3.11, where a modern SDN controller replaces the user and management systems previously used.



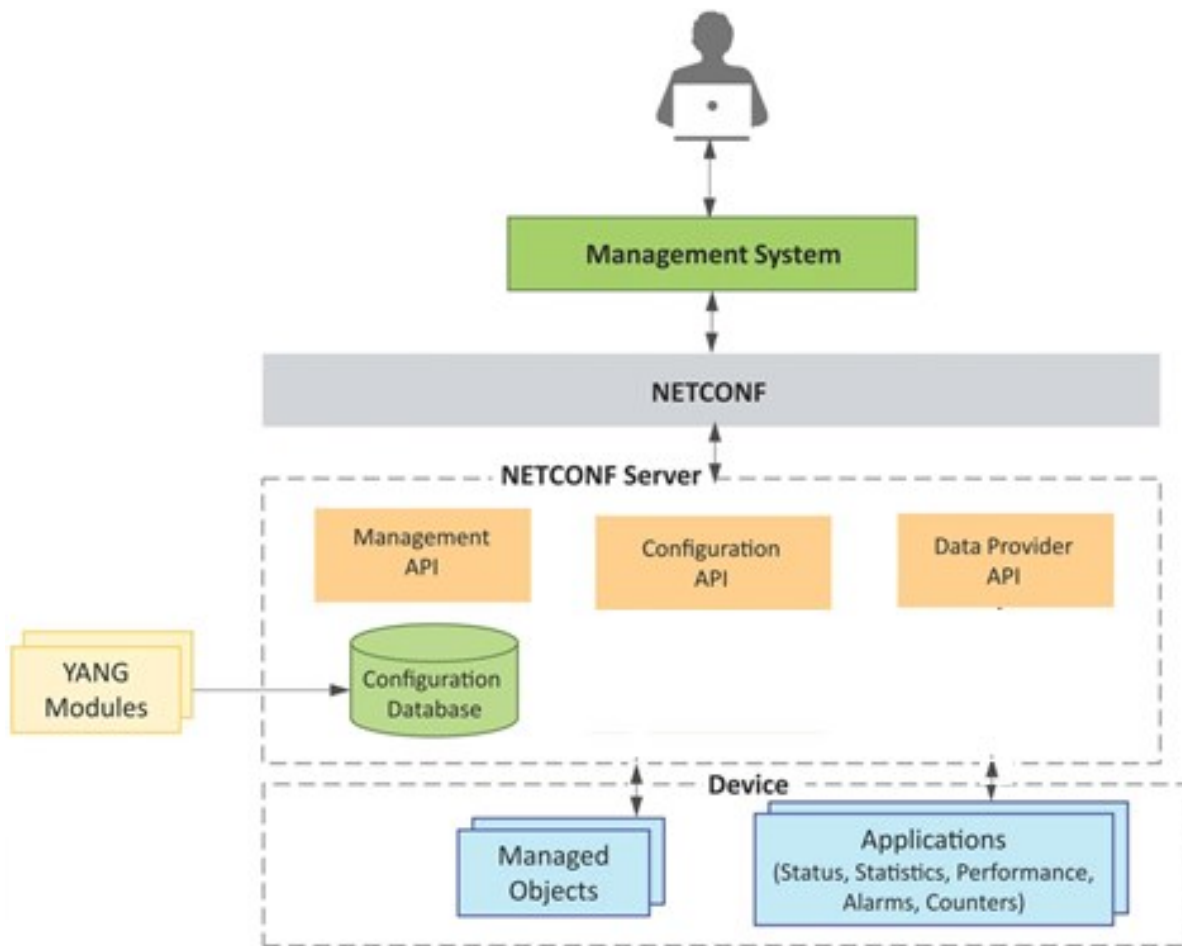


Figure 3.11: NETCONF system architecture²⁷

A NETCONF server is designed to contain an API that enables OS-level applications to access and update parameters stored within its YANG-defined datastore. It also allows applications to contain a call-back function which executes when the parameter it is monitoring in the datastore is updated from a remote device. To achieve SDN control of THz links with the NETCONF protocol, a NETCONF server and data handling service needs to be installed on the OS of the network device (in this example, a switch). “Sysrepo” is an application which provides this service as a complete solution for network devices operating on a Linux-based OS²⁸. An architecture diagram for this solution can be seen in Figure 3. 12, showing the process of accessing and configuring a THz SFP module from the SDN controller on the left to right.

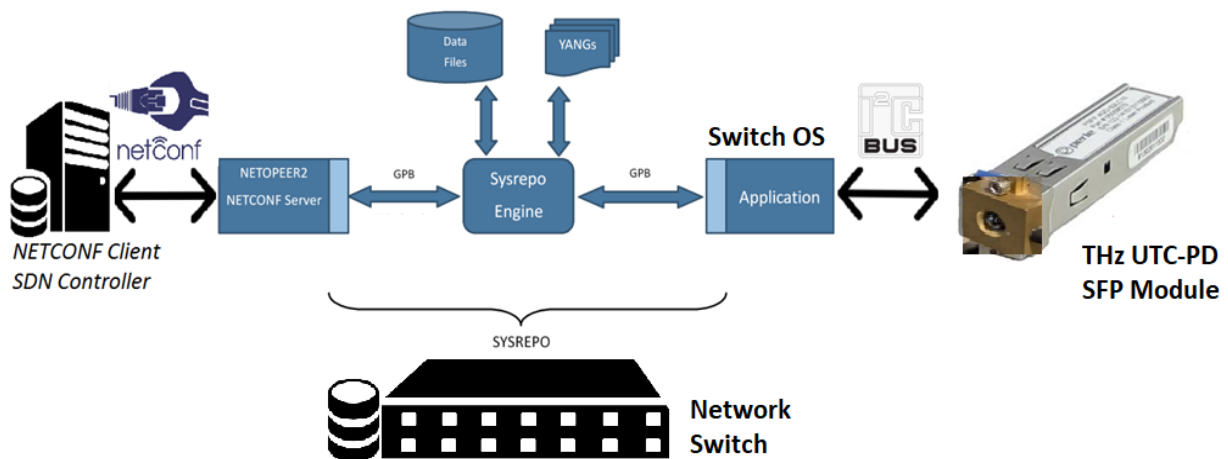


Figure 3. 12: NETCONF Architecture to Implement THz links



The first step for integrating control of THz links with the Sysrepo engine is development of an application which can access and modify the THz EEPROM registers using the I²C bus. This can be achieved using the C programming language, as it contains native driver headers to utilize the I²C bus and is also the API used between Sysrepo and the host OS. Once application control of the THz device is verified, a YANG module needs to be created that is based on the EEPROM register definitions in the example SFF specification. An example of this YANG module can be seen in Figure 3.13, which defines the type and size of the parameters utilized by the THz module.

```

module sysrepo-thz-system
{
  yang-version 1;
  namespace "urn:ietf:params:xml:ns:yang:sysrepo-thz-system";
  prefix sysrepo-thz;
  description "Sysrepo THz Wireless SFP YANG Module";
  container thz
  {
    description "Container for THz parameters";
    leaf port
    {
      type uint8;
      description "Which SFP port on the switch will be accessed";
    }
    leaf frequency
    {
      type uint16;
      description "THz link frequency value";
    }
    leaf width
    {
      type uint16;
      description "THz Channel Width";
    }
    leaf txpower
    {
      type uint32;
      description "THz Transmit Power";
    }
    leaf txbitrate
    {
      type uint32;
      description "THz Current Bitrate";
    }
    leaf ber
    {
      type uint8;
      description "Current Bit Error Rate";
    }
    leaf modulation
    {
      type uint8;
      description "Current Modulation Scheme";
    }
  }
}

```

Figure 3.13: Example YANG model containing THz link parameters



Once this YANG module is installed in the Sysrepo datastore, the C application can be modified to connect to the Sysrepo service, and independent functions can be defined for each leaf parameter value found within the YANG module. The C application can periodically update the datastore with new information taken from the THz EEPROM or can wait for the datastore to be updated remotely, signifying a desired configuration change and writing the new value in the datastore to the THz EEPROM. The function performed by the C application can be tailored to user requirements and use cases. Similar to previous protocols, a C translation function can exist in the application to convert between the YANG values and the EEPROM values. Once the YANG module is installed and the C application is connected to Sysrepo, the THz SFP module is available to be configured from an SDN controller. The controller can connect to the NETCONF server running on the switch and using the THz YANG module, read or write to parameters within that module to control the THz link. A major advantage to this implementation compared to OpenFlow or P4 is that it is backwards-compatible with existing network devices. Unlike those protocols where a new specification or architecture model is required, the NETCONF system is fully software-based and as such is implementable on any device using a Linux OS kernel and containing an I²C bus at any time. Like P4 it can also be updated to enable new THz features on any device as they become available, only requiring an updated YANG module and C code. One major disadvantage to this implementation is the lack of interoperability with packet forwarding ASIC's. This means traditional networking or the use of a second SDN protocol is required by the SDN controller to achieve full control of all network devices. This increases the complexity of programming VNF's for the SDN controller, as now the application needs to be written to operate with two different SDN API's at once. This trade-off could be seen as acceptable however to enable SDN control of THz links on legacy network devices.

3.5 Summary

In summary, this section of the deliverable has defined how to implement SDN control of THz links in datacentre networks based on an extension of the SFF specification for SFP modules. Abstraction of this specification to be configurable via an SDN controller was then described for three common SDN API's OpenFlow, P4, and NETCONF. As shown in the description of this API's there are a number of advantages and disadvantages to the implementation of each, thus the protocol chosen to be implemented may depend on the requirements of the THz link user and their KPI's. Table 2 below shows a summary comparison of some important KPI's which should be considered when implementing SDN control of THz links. Real-time Features is an important metric for THz wireless links and describes functions which require a very high processing speed when a configuration change is made. THz beam steering can be considered a feature which requires this in order to implement advanced wireless network techniques such as Time-Division Multiple-Access (TDMA). Such features are not possible with the NETCONF protocol as the control plane is not capable of changing the configuration at the rate required. While possible with the OpenFlow protocol, its fixed-function data plane limits the capabilities of real time features. This makes P4 the protocol of choice for development of advanced THz network features and should be considered as the long-term choice for future THz SDN development.

KPI	OpenFlow	P4	NETCONF
Complexity	Medium	High	Low
Development Time	Medium	High	Low
Upgradeability	Low	High	High
Data Plane Control	Fixed	Programmable	None
THz Control Speed	Fast	Fastest	Slow
Backwards Compatible	No	No	Yes
Real-time Features	Limited	Supported	None

Table 2: Comparison of KPI's for implementing SDN control of THz links



4 THz Network Function Virtualization

With abstraction of THz links parameters to the network layer achieved, VNF's utilizing these parameters must be created to enable effective control of these THz links at the network layer. The first step in this process is to choose an SDN controller to test implementation of THz VNF's with. An SDN controller known as Ryu was chosen for this task. It is an SDN controller based on the Python programming language and contains several functions and features useful for SDN development²⁹. Ryu's software architecture can be seen in Figure 4.1, showing the data plane network switches at the bottom connecting to Ryu via OpenFlow or another protocol. Ryu contains a number of built-in apps to provide basic network layer functions such as topology discovery, firewalls, and more. A user could choose to expand the built-in apps in Ryu to support THz links or develop an independent VNF which interfaces with the controller, denoted by the SDN apps in this diagram.

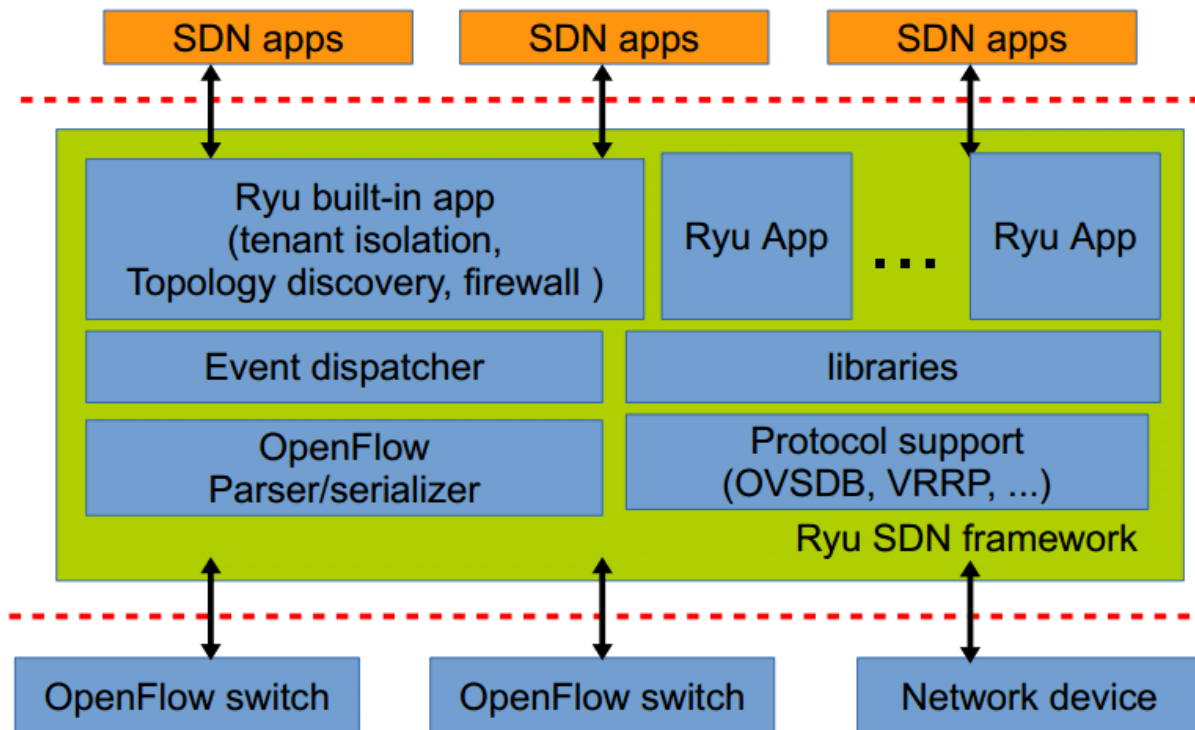


Figure 4.1: Ryu SDN Controller Architecture²⁹

4.1 Topology Discovery

One potentially major change to the way SDN will operate with THz wireless links is in the method of topology discovery. Traditionally, a wired point-to-point link only has one possible destination. The Link Layer Discovery Protocol (LLDP) exploits this principal to determine what devices are connected to each other on the networks data plane. An SDN controller sends a command to a connected switch to perform LLDP on a specific port. This port sends an LLDP packet out of its interface, which is received by a port on another switch. When this switch detects it received an LLDP packet it responds with its own unique identifier, and the port on which it received the packet contained in a metadata format. With future THz links capable of beam steering, point-to-multipoint communication between devices becomes possible. In order for LLDP to support this new form of communication, a number of changes have to be made. The topology discovery function contained within the SDN controller must be modified in order to support the concept of point-to-multipoint communication. This can be achieved by programming the controller and topology discovery function to maintain metadata tables which can contain more than a single instance of an LLDP respondent. This table can later be used to determine what THz-enabled switches are within range to establish a connection with each other, including if more than one THz link is available between a set of switches.



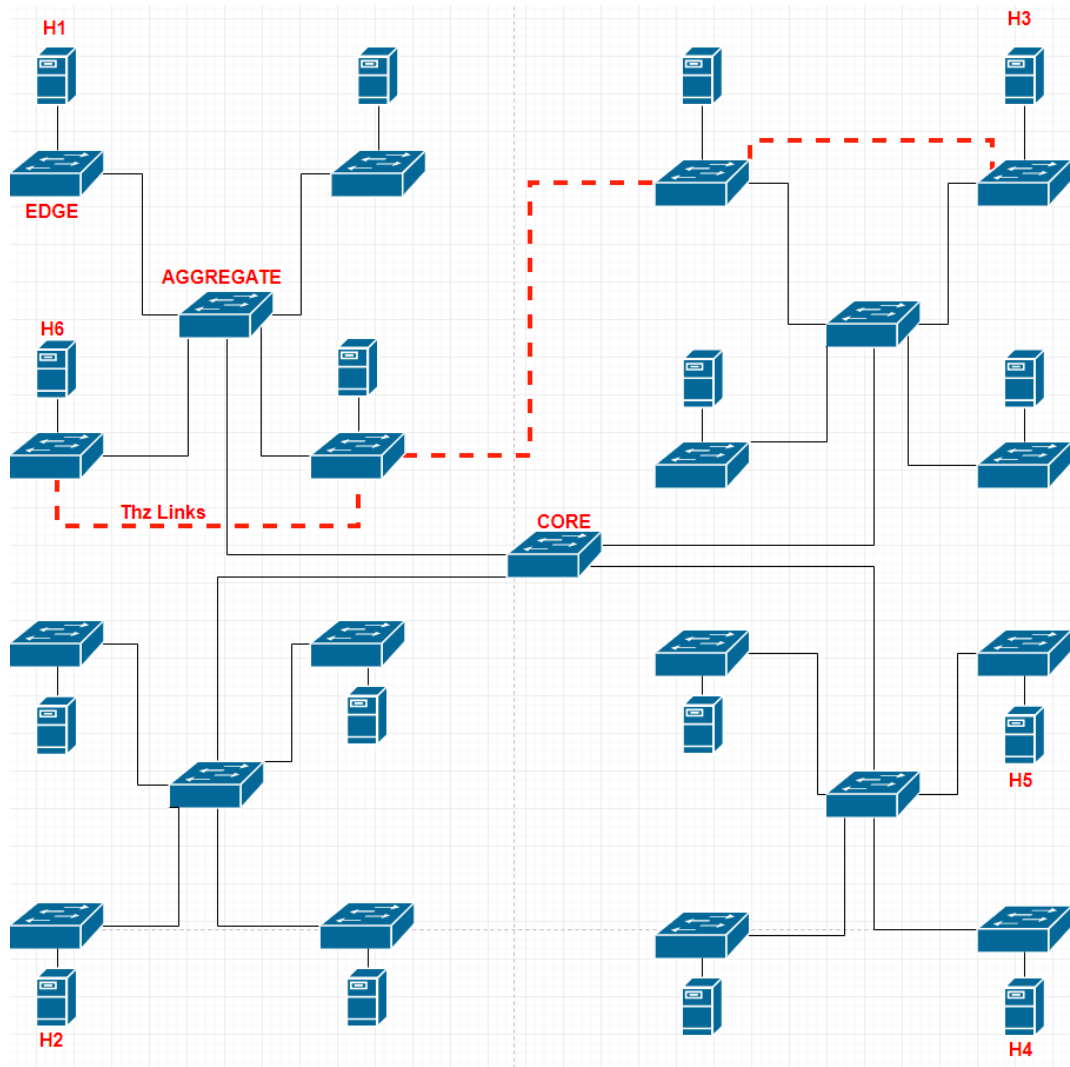


Figure 4.3: Network topology for THz NFV testing

The first step in improving utilization of THz links is the development of a routing algorithm which considers various THz parameters when calculating its routing decisions. In this example all THz links in the demonstration are set to a bandwidth of 10Gbps, while all wired links are set to a bandwidth of 1Gbps. The reason for this is to design the THz routing algorithm to initially favour routing traffic between H3 and H6 over the THz links. This routing algorithm is quite simple and is based on creating a path cost value for various potential routes between H3 and H6. The switch begins the route calculation with the switch connected to H3 and calculates the cost of the available paths based on the available link bandwidth. The available bandwidth cost is measured based on current throughput load of the link versus its maximum bandwidth. In this case a new parameter is added when calculating the cost of a route path: the THz transmission power. If the algorithm detects that the connection between the current hop and the next is a THz link, it will adjust the cost associated with path based on the current THz transmission power of the device. An excerpt from this algorithm can be seen in Figure 4.4.

```
def find_path_cost(self,path):
    path_cost = []
    for i in range(len(path) - 1):
        port1 = self.neigh[path[i]][path[i + 1]]
        bandwidth_between_two_nodes = self.bw[path[i]][port1]
        if (self.thz[path[i]][port1]):
            path_cost.append(THZ_POWER / bandwidth_between_two_nodes)
        path_cost.append(bandwidth_between_two_nodes)
    return sum(path_cost)
```

Figure 4.4: Excerpt of Ryu Thz Routing algorithm



As previously mentioned, the initial state of the routing algorithm should favour using the 10Gbps THz links. During link testing however, the THz transmission power is set to slowly lose power over time. This should eventually cause the routing algorithm to switch the data flow between H3 and H6 over to the 1Gbps when the path cost for the THz link exceeds that of the wired link. In this example an available bandwidth of 1Gbps between nodes is given a cost of 10, while a 10Gbps link is given a cost of 1. A THz transmission power of 1 micro-watt (μW) is given a cost of 1, with a transmission power of $0.1 \mu\text{W}$ given a proportional cost of 10. This routing algorithm runs continuously within the SDN controller as a background process. This means as the transmission power of the THz device weakens over time, the SDN routing algorithm should automatically switch the flow of traffic between H3 and H6 from the THz link to the wired link once the cost of the THz link exceeds the cost of the wired link. If the controller performs this action successfully, it signifies the algorithm is capable of enhanced routing, load-balancing, and fail-over capabilities. While this example algorithm is simple in nature, it provides a proof of concept for the successful implementation of future and further advanced THz VNF's. The results of the test can be seen in figures Figure 4.5. A TCP is sending the maximum amount of data possible between H3 and H6. During this test THz transmission power is slowly weakening, until the route cost for the THz link exceeds that of the wired link at $t=14$ seconds. At this point the controller successfully re-routes the traffic to be sent over the wired link at 1Gbps.

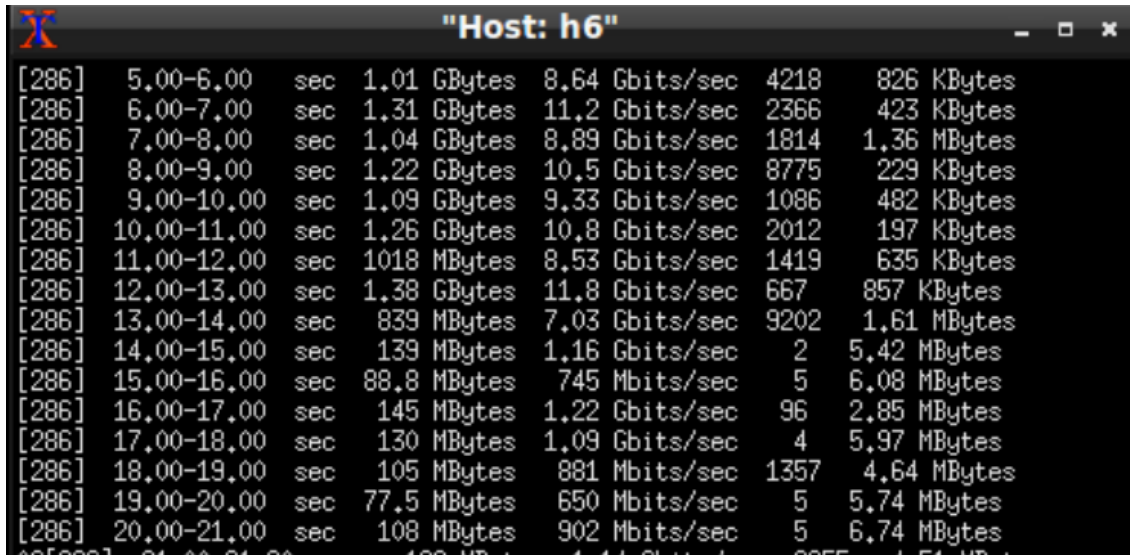


Figure 4.5: TCP test showing re-routing of THz traffic through loss of throughput

It is important for network reliability purposes that this path change and fail-over procedure occurs with minimal delay. Figure 4.6 contains packet capture data which shows the service interruption delay caused by the path change. In this example, network connectivity was restored in 310ms, as noted by the maximum TCP delta time. This time represents the value of time in seconds which passed between this packet being received and the previous one in the TCP sequence.

Time	Source	Destination	Protocol	Info	Delta Time ^
0.469839815	fe:45:dc...	CayeeCom...	LLDP	TTL = 4919 Syste...	0.454461193
0.779967703	10.0.0.6	10.0.0.3	TCP	[TCP Spurious Re...	0.310127888
0.004055260	10.0.0.6	10.0.0.3	TCP	[TCP Previous se...	0.003415025
0.015378622	10.0.0.3	10.0.0.6	TCP	[TCP ACKed unsee...	0.001243542
0.010547433	10.0.0.6	10.0.0.3	TCP	58236 → 5201 [AC...	0.000745424
0.000613535	10.0.0.6	10.0.0.3	TCP	[TCP Previous se...	0.000593535
0.780531362	10.0.0.6	10.0.0.3	TCP	[TCP Previous se...	0.000548579
0.001165694	10.0.0.3	10.0.0.6	TCP	5201 → 58236 [AC...	0.000538799
0.001134234	10.0.0.3	10.0.0.6	TCP	5201 → 58236 [AC...	0.000508209
0.781027424	10.0.0.3	10.0.0.6	TCP	5201 → 58236 [AC...	0.000482992

Figure 4.6: Latency test result for THz fail-over function



5 Conclusions

Based on the investigations performed and results gathered in this deliverable, we can conclude that full integration of Terahertz-based wireless devices with software-defined datacentre networks is possible. A number of methods have been found to implement SDN control of THz links and their introduction into a datacentre environment could cause a dramatic change in the architecture and behaviour of networks that use them. With the bitrates offered by THz devices continuing to increase, their improved flexibility combined with their convenience and potentially reduced physical footprint make them a very lucrative technology in space-saving datacentre network designs. THz technology features such as beam steering have the potential to enable radically different network architectures in the future, with the future potential features and performance of this technology in infrastructure networks exceeding that of both wired and other wireless technologies.

This deliverable satisfies a number of requirements defined in TERAPOD WP2, these are: REQ-F08 to F-13, F-20, F-23, and N06 to N10.



References

- ¹ Kreutz, Diego, et al. "Software-defined networking: A comprehensive survey." *Proceedings of the IEEE* 103.1 (2014): 14-76.
- ² McKeown, Nick, et al. "OpenFlow: enabling innovation in campus networks." *ACM SIGCOMM Computer Communication Review* 38.2 (2008): 69-74.
- ³ Bosshart, Pat, et al. "P4: Programming protocol-independent packet processors." *ACM SIGCOMM Computer Communication Review* 44.3 (2014): 87-95.
- ⁴ Enns, Rob, Martin Bjorklund, and Juergen Schoenwaelder. *NETCONF configuration protocol*. RFC 4741, December, 2006.
- ⁵ Mijumbi, Rashid, et al. "Network function virtualization: State-of-the-art and research challenges." *IEEE Communications surveys & tutorials* 18.1 (2015): 236-262.
- ⁶ Akyildiz, Ian F., Josep Miquel Jornet, and Chong Han. "Terahertz band: Next frontier for wireless communications." *Physical Communication* 12 (2014): 16-32.
- ⁷ Urata, Ryohei, et al. "Datacenter interconnect and networking: From evolution to holistic revolution." *2017 Optical Fiber Communications Conference and Exhibition (OFC)*. IEEE, 2017.
- ⁸ Bellalta, Boris. "IEEE 802.11 ax: High-efficiency WLANs." *IEEE Wireless Communications* 23.1 (2016): 38-46.
- ⁹ Nagatsuma, Tadao, Guillaume Ducournau, and Cyril C. Renaud. "Advances in terahertz communications accelerated by photonics." *Nature Photonics* 10.6 (2016): 371.
- ¹⁰ Chinni, V. K., et al. "Single-channel 100 Gbit/s transmission using III–V UTC-PDs for future IEEE 802.15. 3d wireless links in the 300 GHz band." *Electronics Letters* 54.10 (2018): 638-640.
- ¹¹ iPerf3, *Tool to perform traffic generation and network performance analysis*, URL: <https://iperf.fr/>
- ¹² Wireshark, *Tool to capture network traffic and perform traffic analytics*, URL: <https://www.wireshark.org/>
- ¹³ Mininet, *Network emulation tool for prototyping Software Defined Networks*, URL: <https://github.com/mininet/mininet>
- ¹⁴ Ayala, Kenneth J. *The 8051 microcontroller*. Cengage Learning, 2005.
- ¹⁵ Wong, Tom Sheau Tung, Adrianus Van Haasteren, and Tze Wei Lim. "Small form factor pluggable (SFP) optical transceiver module and method." U.S. Patent No. 7,824,113. 2 Nov. 2010.
- ¹⁶ Wang, Chloe. "Gigabit Ethernet Physical Layer in the Optical Modules - Fiber Transceiver Solution." *Fiber Transceiver Solution*, 17 Mar. 2014, www.fiber-optic-transceiver-module.com/gigabit-ethernet-physical-layer-in-the-optical-modules.html
- ¹⁷ Wikipedia Contributors. "I2C." Wikipedia, Wikimedia Foundation, 2020, <https://en.wikipedia.org/wiki/I%C2%B2C>
- ¹⁸ Bilke, Kevin. "Analog Switch Multiplexes SFP Modules - Maxim Integrated." *Maxim Integrated.Com*, 2010, www.maximintegrated.com/en/design/technical-documents/app-notes/4/4552.html
- ¹⁹ Linux Foundation, "Implementing I2C Device Drivers — The Linux Kernel Documentation." *Kernel.Org*, 2020, www.kernel.org/doc/html/latest/i2c/writing-clients.html
- ²⁰ SFF Committee. "Sff-8472 specification for diagnostic monitoring interface for optical transceivers.", <https://members.snia.org/document/dl/25916> (2018).
- ²¹ SFF Committee. "Sff-8690 specification for Tunable SFP+ Memory Map for ITU Frequencies.", <https://members.snia.org/document/dl/25977> (2013).
- ²² IEEE Standard for High Data Rate Wireless Multi-Media Networks--Amendment 2: 100 Gb/s Wireless Switched Point-to-Point Physical Layer," in *IEEE Std 802.15.3d-2017 (Amendment to IEEE Std 802.15.3-2016 as amended by IEEE Std 802.15.3e-2017)*, vol., no., pp.1-55, 18 Oct. 2017, doi: 10.1109/IEEESTD.2017.8066476.



-
- ²³ Open Networking Foundation, “*OpenFlow 1.5 Implementation - ONOS - Wiki*.” Onosproject.Org, 2016, <https://wiki.onosproject.org/display/ONOS/OpenFlow+1.5+Implementation>
- ²⁴ Open Networking Foundation, “*OpenFlow Switch Specification Version 1.4.0 (Wire Protocol 0x05)*”. 2013, <https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf>
- ²⁵ Bosshart, Pat, et al. "P4: Programming protocol-independent packet processors." ACM SIGCOMM Computer Communication Review 44.3 (2014): 87-95.
- ²⁶ The P4 Language Consortium. “P4~16~ Language Specification.” P4.Org, 2020, <https://p4.org/p4-spec/docs/P4-16-working-spec.html>
- ²⁷ Bahga, Arshdeep et al. “*Software Defined Things in Manufacturing Networks. Journal of Software Engineering and Applications*.”, (2016), 09. 425-438. 10.4236/jsea.2016.99028
- ²⁸ sysrepo. “Sysrepo” GitHub, 7 May 2020, <https://github.com/sysrepo/sysrepo>
- ²⁹ FaucetSDN. “Ryu.” GitHub, 22 May 2020, <https://github.com/faucetsdn/ryu>

